Proceedings of HLT-NAACL 2003
Main Papers , pp. 134-141
Edmonton, May-June 2003

Shallow Parsing with Conditional Random Fields

Fei Sha and **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104
(feisha|pereira)@cis.upenn.edu

Abstract

Conditional random fields for sequence labeling offer advantages over both generative models like HMMs and classifiers applied at each sequence position. Among sequence labeling tasks in language processing, shallow parsing has received much attention, with the development of standard evaluation datasets and extensive comparison among methods. We show here how to train a conditional random field to achieve performance as good as any reported base noun-phrase chunking method on the CoNLL task, and better than any reported single model. Improved training methods based on modern optimization algorithms were critical in achieving these results. We present extensive comparisons between models and training methods that confirm and strengthen previous results on shallow parsing and training methods for maximum-entropy models.

1 Introduction

Sequence analysis tasks in language and biology are often described as mappings from input sequences to sequences of labels encoding the analysis. In language processing, examples of such tasks include part-of-speech tagging, named-entity recognition, and the task we shall focus on here, shallow parsing. Shallow parsing identifies the non-recursive cores of various phrase types in text, possibly as a precursor to full parsing or information extraction (Abney, 1991). The paradigmatic shallow-parsing problem is *NP chunking*, which finds the non-recursive cores of noun phrases called *base NPs*. The pioneering work of Ramshaw and Marcus (1995) introduced NP chunking as a machine-learning problem, with standard datasets and evaluation metrics. The task was extended to additional phrase types for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000), which is now the standard evaluation task for shallow parsing.

Most previous work used two main machine-learning approaches to sequence labeling. The first approach relies on k-order generative probabilistic models of paired input sequences and label sequences, for instance hidden Markov models (HMMs) (Freitag and McCallum, 2000; Kupiec, 1992) or multilevel Markov models (Bikel et al., 1999). The second approach views the sequence labeling problem as a sequence of classification problems, one for each of the labels in the sequence. The classification result at each position may depend on the whole input and on the previous k classifications. [1]

The generative approach provides well-understood training and decoding algorithms for HMMs and more general graphical models. However, effective generative models require stringent conditional independence assumptions. For instance, it is not practical to make the label at a given position depend on a window on the input sequence as well as the surrounding labels, since the inference problem for the corresponding graphical model would be intractable. Non-independent features of the inputs, such as capitalization, suffixes, and surrounding words, are important in dealing with words unseen in training, but they are difficult to represent in generative models.

The sequential classification approach can handle many correlated features, as demonstrated in work on maximum-entropy (McCallum et al., 2000; Ratnaparkhi, 1996) and a variety of other linear classifiers, including winnow (Punyakanok and Roth, 2001), AdaBoost (Abney et al., 1999), and support-vector machines (Kudo and Matsumoto, 2001). Furthermore, they are trained to minimize some function related to labeling error, leading to smaller error in practice if enough training data are available. In contrast, generative models are trained to maximize the joint probability of the training data, which is

[1] Ramshaw and Marcus (1995) used transformation-based learning (Brill, 1995), which for the present purposes can be tought of as a classification-based method.

not as closely tied to the accuracy metrics of interest if the actual data was not generated by the model, as is always the case in practice.

However, since sequential classifiers are trained to make the best local decision, unlike generative models they cannot trade off decisions at different positions against each other. In other words, sequential classifiers are myopic about the impact of their current decision on later decisions (Bottou, 1991; Lafferty et al., 2001). This forced the best sequential classifier systems to resort to heuristic combinations of forward-moving and backward-moving sequential classifiers (Kudo and Matsumoto, 2001).

Conditional random fields (CRFs) bring together the best of generative and classification models. Like classification models, they can accommodate many statistically correlated features of the inputs, and they are trained discriminatively. But like generative models, they can trade off decisions at different sequence positions to obtain a globally optimal labeling. Lafferty et al. (2001) showed that CRFs beat related classification models as well as HMMs on synthetic data and on a part-of-speech tagging task.

In the present work, we show that CRFs beat all reported single-model NP chunking results on the standard evaluation dataset, and are statistically indistinguishable from the previous best performer, a voting arrangement of 24 forward- and backward-looking support-vector classifiers (Kudo and Matsumoto, 2001). To obtain these results, we had to abandon the original iterative scaling CRF training algorithm for convex optimization algorithms with better convergence properties. We provide detailed comparisons between training methods.

The generalized perceptron proposed by Collins (2002) is closely related to CRFs, but the best CRF training methods seem to have a slight edge over the generalized perceptron.

2 Conditional Random Fields

We focus here on conditional random fields on sequences, although the notion can be used more generally (Lafferty et al., 2001; Taskar et al., 2002). Such CRFs define conditional probability distributions $p(\boldsymbol{Y}|\boldsymbol{X})$ of label sequences given input sequences. We assume that the random variable sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ have the same length, and use $\boldsymbol{x} = x_1 \cdots x_n$ and $\boldsymbol{y} = y_1 \cdots y_n$ for the generic input sequence and label sequence, respectively.

A CRF on $(\boldsymbol{X}, \boldsymbol{Y})$ is specified by a vector $\boldsymbol{f}$ of *local features* and a corresponding *weight vector* $\boldsymbol{\lambda}$. Each local feature is either a *state feature* $s(y, \boldsymbol{x}, i)$ or a *transition feature* $t(y, y', \boldsymbol{x}, i)$, where y, y' are labels, $\boldsymbol{x}$ an input sequence, and i an input position. To make the notation more uniform, we also write

$$\begin{aligned} s(y, y', \boldsymbol{x}, i) &= s(y', \boldsymbol{x}, i) \\ s(\boldsymbol{y}, \boldsymbol{x}, i) &= s(y_i, \boldsymbol{x}, i) \\ t(\boldsymbol{y}, \boldsymbol{x}, i) &= \begin{cases} t(y_{i-1}, y_i, \boldsymbol{x}, i) & i > 1 \\ 0 & i = 1 \end{cases} \end{aligned}$$

for any state feature s and transition feature t. Typically, features depend on the inputs around the given position, although they may also depend on global properties of the input, or be non-zero only at some positions, for instance features that pick out the first or last labels.

The CRF's *global feature vector* for input sequence $\boldsymbol{x}$ and label sequence $\boldsymbol{y}$ is given by

$$\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}) = \sum_i \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}, i)$$

where i ranges over input positions. The conditional probability distribution defined by the CRF is then

$$p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{X}) = \frac{\exp \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{Y}, \boldsymbol{X})}{Z_{\boldsymbol{\lambda}}(\boldsymbol{X})} \tag{1}$$

where

$$Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})$$

Any positive conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X})$ that obeys the *Markov property*

$$p(Y_i|\{Y_j\}_{j \neq i}, \boldsymbol{X}) = p(Y_i|Y_{i-1}, Y_{i+1}, \boldsymbol{X})$$

can be written in the form (1) for appropriate choice of feature functions and weight vector (Hammersley and Clifford, 1971).

The most probable label sequence for input sequence $\boldsymbol{x}$ is

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \arg\max_{\boldsymbol{y}} \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})$$

because $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$ does not depend on $\boldsymbol{y}$. $\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})$ decomposes into a sum of terms for consecutive pairs of labels, so the most likely $\boldsymbol{y}$ can be found with the Viterbi algorithm.

We train a CRF by maximizing the log-likelihood of a given training set $T = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^N$, which we assume fixed for the rest of this section:

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\lambda}} &= \textstyle\sum_k \log p_{\boldsymbol{\lambda}}(\boldsymbol{y}_k|\boldsymbol{x}_k) \\ &= \textstyle\sum_k [\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}_k, \boldsymbol{x}_k) - \log Z_{\boldsymbol{\lambda}}(\boldsymbol{x}_k)] \end{aligned}$$

To perform this optimization, we seek the zero of the gradient

$$\nabla \mathcal{L}_{\boldsymbol{\lambda}} = \sum_k \left[\boldsymbol{F}(\boldsymbol{y}_k, \boldsymbol{x}_k) - E_{p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{x}_k)} \boldsymbol{F}(\boldsymbol{Y}, \boldsymbol{x}_k)\right] \tag{2}$$

In words, the maximum of the training data likelihood is reached when the empirical average of the global feature vector equals its model expectation. The expectation $E_{p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{x})}\boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x})$ can be computed efficiently using a variant of the forward-backward algorithm. For a given $\boldsymbol{x}$, define the *transition matrix* for position i as

$$M_i[y,y'] = \exp \boldsymbol{\lambda} \cdot \boldsymbol{f}(y,y',\boldsymbol{x},i)$$

Let f be any local feature, $f_i[y,y'] = f(y,y',\boldsymbol{x},i)$, $F(\boldsymbol{y},\boldsymbol{x}) = \sum_i f(y_{i-1},y_i,\boldsymbol{x},i)$, and let $*$ denote component-wise matrix product. Then

$$\begin{aligned} E_{p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{x})}F(\boldsymbol{Y},\boldsymbol{x}) &= \sum_{\boldsymbol{y}} p_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x})F(\boldsymbol{y},\boldsymbol{x}) \\ &= \sum_i \frac{\boldsymbol{\alpha}_{i-1}(f_i * M_i)\boldsymbol{\beta}_i^\top}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} \\ Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) &= \boldsymbol{\alpha}_n \cdot \mathbf{1}^\top \end{aligned}$$

where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ the forward and backward state-cost vectors defined by

$$\begin{aligned} \boldsymbol{\alpha}_i &= \begin{cases} \boldsymbol{\alpha}_{i-1}M_i & 0 < i \leq n \\ \mathbf{1} & i = 0 \end{cases} \\ \boldsymbol{\beta}_i^\top &= \begin{cases} M_{i+1}\boldsymbol{\beta}_{i+1}^\top & 1 \leq i < n \\ \mathbf{1} & i = n \end{cases} \end{aligned}$$

Therefore, we can use a forward pass to compute the $\boldsymbol{\alpha}_i$ and a backward bass to compute the $\boldsymbol{\beta}_i$ and accumulate the feature expectations.

To avoid overfitting, we penalize the likelihood with a spherical Gaussian weight prior (Chen and Rosenfeld, 1999):

$$\begin{aligned} \mathcal{L}'_{\boldsymbol{\lambda}} = &\sum_k \left[\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}_k,\boldsymbol{x}_k) - \log Z_{\boldsymbol{\lambda}}(\boldsymbol{x}_k)\right] \\ &- \frac{\|\boldsymbol{\lambda}\|^2}{2\sigma^2} + \text{const} \end{aligned}$$

with gradient

$$\begin{aligned} &\nabla \mathcal{L}'_{\boldsymbol{\lambda}} = \\ &\sum_k \left[\boldsymbol{F}(\boldsymbol{y}_k,\boldsymbol{x}_k) - E_{p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{x}_k)}\boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x}_k)\right] - \frac{\boldsymbol{\lambda}}{\sigma^2} \end{aligned}$$

3 Training Methods

Lafferty et al. (2001) used iterative scaling algorithms for CRF training, following earlier work on maximum-entropy models for natural language (Berger et al., 1996; Della Pietra et al., 1997). Those methods are very simple and guaranteed to converge, but as Minka (2001) and Malouf (2002) showed for classification, their convergence is much slower than that of general-purpose convex optimization algorithms when many correlated features are involved. Concurrently with the present work, Wallach (2002) tested conjugate gradient and second-order methods for CRF training, showing significant training speed advantages over iterative scaling on a small shallow parsing problem. Our work shows that preconditioned conjugate-gradient (CG) (Shewchuk, 1994) or limited-memory quasi-Newton (L-BFGS) (Nocedal and Wright, 1999) perform comparably on very large problems (around 3.8 million features). We compare those algorithms to generalized iterative scaling (GIS) (Darroch and Ratcliff, 1972), non-preconditioned CG, and voted perceptron training (Collins, 2002). All algorithms except voted perceptron maximize the penalized log-likelihood: $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \mathcal{L}'_{\boldsymbol{\lambda}}$. However, for ease of exposition, this discussion of training methods uses the unpenalized log-likelihood $\mathcal{L}_{\boldsymbol{\lambda}}$.

3.1 Preconditioned Conjugate Gradient

Conjugate-gradient (CG) methods have been shown to be very effective in linear and non-linear optimization (Shewchuk, 1994). Instead of searching along the gradient, conjugate gradient searches along a carefully chosen linear combination of the gradient and the previous search direction.

CG methods can be accelerated by linearly transforming the variables with *preconditioner* (Nocedal and Wright, 1999; Shewchuk, 1994). The purpose of the preconditioner is to improve the condition number of the quadratic form that locally approximates the objective function, so the inverse of Hessian is reasonable preconditioner. However, this is not applicable to CRFs for two reasons. First, the size of the Hessian is $\dim(\boldsymbol{\lambda})^2$, leading to unacceptable space and time requirements for the inversion. In such situations, it is common to use instead the (inverse of) the diagonal of the Hessian. However in our case the Hessian has the form

$$\begin{aligned} H_{\boldsymbol{\lambda}} &\stackrel{\text{def}}{=} \nabla^2 \mathcal{L}_{\boldsymbol{\lambda}} \\ &= -\sum_k \{E\left[\boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x}_k) \times \boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x}_k)\right] \\ &\qquad -E\boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x}_k) \times E\boldsymbol{F}(\boldsymbol{Y},\boldsymbol{x}_k)\} \end{aligned}$$

where the expectations are taken with respect to $p_{\boldsymbol{\lambda}}(\boldsymbol{Y}|\boldsymbol{x}_k)$. Therefore, every Hessian element, including the diagonal ones, involve the expectation of a product of global feature values. Unfortunately, computing those expectations is quadratic on sequence length, as the forward-backward algorithm can only compute expectations of quantities that are additive along label sequences.

We solve both problems by discarding the off-diagonal terms and approximating expectation of the square of a global feature by the expectation of the sum of squares of the corresponding local features at each position. The ap-

proximated diagonal term H_f for feature f has the form

$$H_f = Ef(\boldsymbol{Y}, \boldsymbol{x}_k)^2 - \sum_i \left(\sum_{y,y'} \frac{M_i[y,y']}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} f(\boldsymbol{Y}, \boldsymbol{x}_k) \right)^2$$

If this approximation is semidefinite, which is trivial to check, its inverse is an excellent preconditioner for early iterations of CG training. However, when the model is close to the maximum, the approximation becomes unstable, which is not surprising since it is based on feature independence assumptions that become invalid as the weights of interaction features move away from zero. Therefore, we disable the preconditioner after a certain number of iterations, determined from held-out data. We call this strategy *mixed* CG training.

3.2 Limited-Memory Quasi-Newton

Newton methods for nonlinear optimization use second-order (curvature) information to find search directions. As discussed in the previous section, it is not practical to obtain exact curvature information for CRF training. Limited-memory BFGS (L-BFGS) is a second-order method that estimates the curvature numerically from previous gradients and updates, avoiding the need for an exact Hessian inverse computation. Compared with preconditioned CG, L-BFGS can also handle large-scale problems but does not require a specialized Hessian approximations. An earlier study indicates that L-BFGS performs well in maximum-entropy classifier training (Malouf, 2002).

There is no theoretical guidance on how much information from previous steps we should keep to obtain sufficiently accurate curvature estimates. In our experiments, storing 3 to 10 pairs of previous gradients and updates worked well, so the extra memory required over preconditioned CG was modest. A more detailed description of this method can be found elsewhere (Nocedal and Wright, 1999).

3.3 Voted Perceptron

Unlike other methods discussed so far, voted perceptron training (Collins, 2002) attempts to minimize the difference between the global feature vector for a training instance and the same feature vector for the best-scoring labeling of that instance according to the current model. More precisely, for each training instance the method computes a weight update

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{F}(\boldsymbol{y}_k, \boldsymbol{x}_k) - \boldsymbol{F}(\hat{\boldsymbol{y}}_k, \boldsymbol{x}_k) \qquad (3)$$

in which $\hat{\boldsymbol{y}}_k$ is the Viterbi path

$$\hat{\boldsymbol{y}}_k = \arg\max_{\boldsymbol{y}} \boldsymbol{\lambda}_t \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}_k)$$

Like the familiar perceptron algorithm, this algorithm repeatedly sweeps over the training instances, updating the weight vector as it considers each instance. Instead of taking just the final weight vector, the voted perceptron algorithm takes the average of the $\boldsymbol{\lambda}_t$. Collins (2002) reported and we confirmed that this averaging reduces overfitting considerably.

4 Shallow Parsing

Figure 1 shows the base NPs in an example sentence. Following Ramshaw and Marcus (1995), the input to the NP chunker consists of the words in a sentence annotated automatically with part-of-speech (POS) tags. The chunker's task is to label each word with a label indicating whether the word is outside a chunk (`O`), starts a chunk (`B`), or continues a chunk (`I`). For example, the tokens in first line of Figure 1 would be labeled `BIIBIIOBOBIIO`.

4.1 Data Preparation

NP chunking results have been reported on two slightly different data sets: the original RM data set of Ramshaw and Marcus (1995), and the modified CoNLL-2000 version of Tjong Kim Sang and Buchholz (2000). Although the chunk tags in the RM and CoNLL-2000 are somewhat different, we found no significant accuracy differences between models trained on these two data sets. Therefore, all our results are reported on the CoNLL-2000 data set. We also used a development test set, provided by Michael Collins, derived from WSJ section 21 tagged with the Brill (1995) POS tagger.

4.2 CRFs for Shallow Parsing

Our chunking CRFs have a second-order Markov dependency between chunk tags. This is easily encoded by making the CRF labels pairs of consecutive chunk tags. That is, the label at position i is $y_i = c_{i-1}c_i$, where c_i is the chunk tag of word i, one of `O`, `B`, or `I`. Since `B` must be used to start a chunk, the label `OI` is impossible. In addition, successive labels are constrained: $y_{i-1} = c_{i-2}c_{i-1}$, $y_i = c_{i-1}c_i$, and $c_0 = \texttt{O}$. These contraints on the model topology are enforced by giving appropriate features a weight of $-\infty$, forcing all the forbidden labelings to have zero probability.

Our choice of features was mainly governed by computing power, since we do not use feature selection and all features are used in training and testing. We use the following factored representation for features

$$f(y_{i-1}, y_i, \boldsymbol{x}, i) = p(\boldsymbol{x}, i) q(y_{i-1}, y_i) \qquad (4)$$

where $p(\boldsymbol{x}, i)$ is a predicate on the input sequence $\boldsymbol{x}$ and current position i and $q(y_{i-1}, y_i)$ is a predicate on pairs of labels. For instance, $p(\boldsymbol{x}, i)$ might be "word at position i is `the`" or "the POS tags at positions $i-1, i$ are

[Rockwell International Corp.] ['s Tulsa unit] said [it] signed [a tentative agreement] extending [its contract] with [Boeing Co.] to provide [structural parts] for [Boeing] ['s 747 jetliners].

Figure 1: NP chunks

$q(y_{i-1}, y_i)$	$p(\boldsymbol{x}, i)$
$y_i = y$ $y_i = y,\ y_{i-1} = y'$ $c(y_i) = c$	true
$y_i = y$ or $c(y_i) = c$	$w_i = w$ $w_{i-1} = w$ $w_{i+1} = w$ $w_{i-2} = w$ $w_{i+2} = w$ $w_{i-1} = w',\ w_i = w$ $w_{i+1} = w',\ w_i = w$ $t_i = t$ $t_{i-1} = t$ $t_{i+1} = t$ $t_{i-2} = t$ $t_{i+2} = t$ $t_{i-1} = t',\ t_i = t$ $t_{i-2} = t',\ t_{i-1} = t$ $t_i = t',\ t_{i+1} = t$ $t_{i+1} = t',\ t_{i+2} = t$ $t_{i-2} = t'',\ t_{i-1} = t',\ t_i = t$ $t_{i-1} = t'',\ t_i = t',\ t_{i+1} = t$ $t_i = t'',\ t_{i+1} = t',\ t_{i+2} = t$

Table 1: Shallow parsing features

DT, NN." Because the label set is finite, such a factoring of $f(y_{i-1}, y_i, \boldsymbol{x}, i)$ is always possible, and it allows each input predicate to be evaluated just once for many features that use it, making it possible to work with millions of features on large training sets.

Table 1 summarizes the feature set. For a given position i, w_i is the word, t_i its POS tag, and y_i its label. For any label $y = c'c$, $c(y) = c$ is the corresponding chunk tag. For example, $c(\texttt{OB}) = \texttt{B}$. The use of chunk tags as well as labels provides a form of backoff from the very small feature counts that may arise in a second-order model, while allowing significant associations between tag pairs and input predicates to be modeled. To save time in some of our experiments, we used only the 820,000 features that are *supported* in the CoNLL training set, that is, the features that are on at least once. For our highest F score, we used the *complete* feature set, around 3.8 million in the CoNLL training set, which contains all the features whose predicate is on at least once in the training set. The complete feature set may in principle perform better because it can place negative weights on transitions that should be discouraged if a given predicate is on.

4.3 Parameter Tuning

As discussed previously, we need a Gaussian weight prior to reduce overfitting. We also need to choose the number of training iterations since we found that the best F score is attained while the log-likelihood is still improving. The reasons for this are not clear, but the Gaussian prior may not be enough to keep the optimization from making weight adjustments that slighly improve training log-likelihood but cause large F score fluctuations. We used the development test set mentioned in Section 4.1 to set the prior and the number of iterations.

4.4 Evaluation Metric

The standard evaluation metrics for a chunker are precision P (fraction of output chunks that exactly match the reference chunks), recall R (fraction of reference chunks returned by the chunker), and their harmonic mean, the F1 score $F_1 = 2 * P * R/(P + R)$ (which we call just F score in what follows). The relationships between F score and labeling error or log-likelihood are not direct, so we report both F score and the other metrics for the models we tested. For comparisons with other reported results we use F score.

4.5 Significance Tests

Ideally, comparisons among chunkers would control for feature sets, data preparation, training and test procedures, and parameter tuning, and estimate the statistical significance of performance differences. Unfortunately, reported results sometimes leave out details needed for accurate comparisons. We report F scores for comparison with previous work, but we also give statistical significance estimates using McNemar's test for those methods that we evaluated directly.

Testing the significance of F scores is tricky because the wrong chunks generated by two chunkers are not directly comparable. Yeh (2000) examined randomized tests for estimating the significance of F scores, and in particular the bootstrap over the test set (Efron and Tibshirani, 1993; Sang, 2002). However, bootstrap variances in preliminary experiments were too high to allow any conclusions, so we used instead a McNemar paired test on labeling disagreements (Gillick and Cox, 1989).

Model	F score
SVM combination (Kudo and Matsumoto, 2001)	94.39%
CRF	94.38%
Generalized winnow (Zhang et al., 2002)	93.89%
Voted perceptron	94.09%
MEMM	93.70%

Table 2: NP chunking F scores

5 Results

All the experiments were performed with our Java implementation of CRFs,designed to handle millions of features, on 1.7 GHz Pentium IV processors with Linux and IBM Java 1.3.0. Minor variants support voted perceptron (Collins, 2002) and MEMMs (McCallum et al., 2000) with the same efficient feature encoding. GIS, CG, and L-BFGS were used to train CRFs and MEMMs.

5.1 F Scores

Table 2 gives representative NP chunking F scores for previous work and for our best model, with the complete set of 3.8 million features. The last row of the table gives the score for an MEMM trained with the mixed CG method using an approximate preconditioner. The published F score for voted perceptron is 93.53% with a different feature set (Collins, 2002). The improved result given here is for the supported feature set; the complete feature set gives a slightly lower score of 94.07%. Zhang et al. (2002) reported a higher F score (94.38%) with generalized winnow using additional linguistic features that were not available to us.

5.2 Convergence Speed

All the results in the rest of this section are for the smaller supported set of 820,000 features. Figures 2a and 2b show how preconditioning helps training convergence. Since each CG iteration involves a line search that may require several forward-backward procedures (typically between 4 and 5 in our experiments), we plot the progress of penalized log-likelihood $\mathcal{L}'_{\boldsymbol{\lambda}}$ with respect to the number of forward-backward evaluations. The objective function increases rapidly, achieving close proximity to the maximum in a few iterations (typically 10). In contrast, GIS training increases $\mathcal{L}'_{\boldsymbol{\lambda}}$ rather slowly, never reaching the value achieved by CG. The relative slowness of iterative scaling is also documented in a recent evaluation of training methods for maximum-entropy classification (Malouf, 2002). In theory, GIS would eventually converge to the $\mathcal{L}'_{\boldsymbol{\lambda}}$ optimum, but in practice convergence may be so slow that $\mathcal{L}'_{\boldsymbol{\lambda}}$ improvements may fall below numerical accuracy, falsely indicating convergence.

training method	time	F score	$\mathcal{L}'_{\boldsymbol{\lambda}}$
Precond. CG	130	94.19%	-2968
Mixed CG	540	94.20%	-2990
Plain CG	648	94.04%	-2967
L-BFGS	84	94.19%	-2948
GIS	3700	93.55%	-5668

Table 3: Runtime for various training methods

null hypothesis	p-value
CRF vs. SVM	0.469
CRF vs. MEMM	0.00109
CRF vs. voted perceptron	0.116
MEMM vs. voted perceptron	0.0734

Table 4: McNemar's tests on labeling disagreements

Mixed CG training converges slightly more slowly than preconditioned CG. On the other hand, CG without preconditioner converges much more slowly than both preconditioned CG and mixed CG training. However, it is still much faster than GIS. We believe that the superior convergence rate of preconditioned CG is due to the use of approximate second-order information. This is confirmed by the performance of L-BFGS, which also uses approximate second-order information.[2]

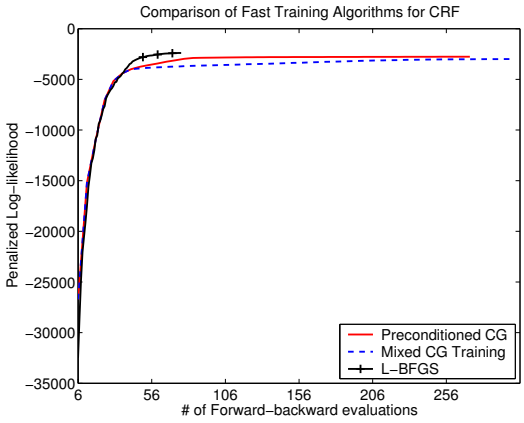
Although there is no direct relationship between F scores and log-likelihood, in these experiments F score tends to follow log-likelihood. Indeed, Figure 3 shows that preconditioned CG training improves test F scores much more rapidly than GIS training.

Table 3 compares run times (in minutes) for reaching a target penalized log-likelihood for various training methods with prior $\sigma = 1.0$. GIS is the only method that failed to reach the target, after 3,700 iterations. We cannot place the voted perceptron in this table, as it does not optimize log-likelihood and does not use a prior. However, it reaches a fairly good F-score above 93% in just two training sweeps, but after that it improves more slowly, to a somewhat lower score, than preconditioned CG training.

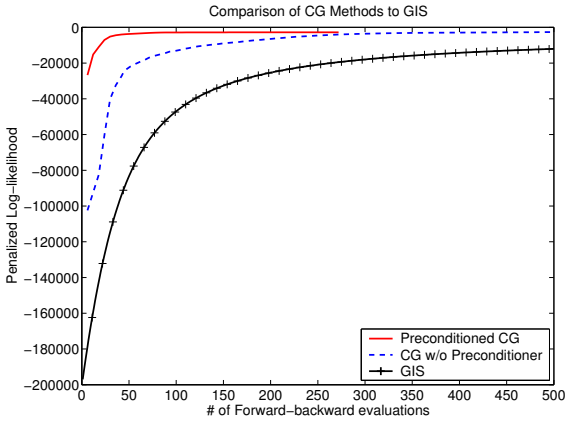
5.3 Labeling Accuracy

The accuracy rate for individual labeling decisions is over-optimistic as an accuracy measure for shallow parsing. For instance, if the chunk `BIIIIIII` is labled as `OIIIIIII`, the labeling accuracy is 87.5%, but recall is 0. However, individual labeling errors provide a more convenient basis for statistical significance tests. One

[2] Although L-BFGS has a slightly higher penalized log-likelihood, its log-likelihood on the data is actually lower than that of preconditioned CG and mixed CG training.



(a) $\mathcal{L}'_{\boldsymbol{\lambda}}$: CG (precond., mixed), L-BFGS



(b) $\mathcal{L}'_{\boldsymbol{\lambda}}$: CG (precond., plain), GIS

Figure 2: Training convergence for various methods

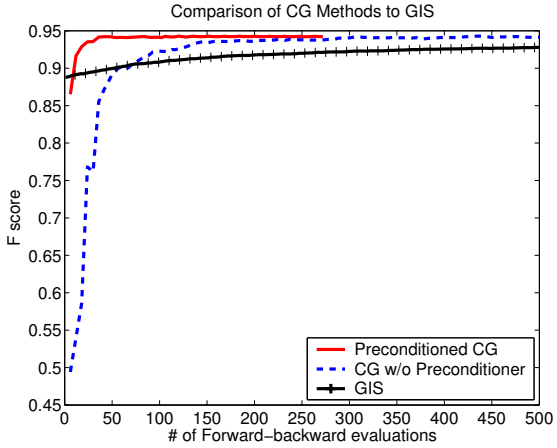


Figure 3: Test F scores *vs.* training time

such test is McNemar test on paired observations (Gillick and Cox, 1989).

With McNemar's test, we compare the correctness of the labeling decisions of two models. The null hypothesis is that the disagreements (correct *vs.* incorrect) are due to chance. Table 4 summarizes the results of tests between the models for which we had labeling decisions. These tests suggest that MEMMs are significantly less accurate, but that there are no significant differences in accuracy among the other models.

6 Conclusions

We have shown that (log-)linear sequence labeling models trained discriminatively with general-purpose optimization methods are a simple, competitive solution to learning shallow parsers. These models combine the best features of generative finite-state models and discriminative (log-)linear classifiers, and do NP chunking as well as or better than "ad hoc" classifier combinations, which were the most accurate approach until now. In a longer version of this work we will also describe shallow parsing results for other phrase types. There is no reason why the same techniques cannot be used equally successfully for the other types or for other related tasks, such as POS tagging or named-entity recognition.

On the machine-learning side, it would be interesting to generalize the ideas of large-margin classification to sequence models, strengthening the results of Collins (2002) and leading to new optimal training algorithms with stronger guarantees against overfitting.

On the application side, (log-)linear parsing models have the potential to supplant the currently dominant lexicalized PCFG models for parsing by allowing much richer feature sets and simpler smoothing, while avoiding the label bias problem that may have hindered earlier classifier-based parsers (Ratnaparkhi, 1997). However, work in that direction has so far addressed only parse reranking (Collins and Duffy, 2002; Riezler et al., 2002). Full discriminative parser training faces significant algorithmic challenges in the relationship between parsing alternatives and feature values (Geman and Johnson, 2002) and in computing feature expectations.

Acknowledgments

John Lafferty and Andrew McCallum worked with the second author on developing CRFs. McCallum helped by the second author implemented the first conjugate-gradient trainer for CRFs, which convinced us that training of large CRFs on large datasets would be practical. Michael Collins helped us reproduce his generalized per-

cepton results and compare his method with ours. Erik Tjong Kim Sang, who has created the best online resources on shallow parsing, helped us with details of the CoNLL-2000 shared task. Taku Kudo provided the output of his SVM chunker for the significance test.

References

S. Abney. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer Academic Publishers, 1991.

S. Abney, R. E. Schapire, and Y. Singer. Boosting applied to tagging and PP attachment. In *Proc. EMNLP-VLC*, New Brunswick, New Jersey, 1999. ACL.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), 1996.

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34: 211–231, 1999.

L. Bottou. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. PhD thesis, Université de Paris XI, 1991.

E. Brill. Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, 21:543–565, 1995.

S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP 2002*. ACL, 2002.

M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. 40th ACL*, 2002.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43 (5):1470–1480, 1972.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE PAMI*, 19(4):380–393, 1997.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proc. AAAI 2000*, 2000.

S. Geman and M. Johnson. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proc. 40th ACL*, 2002.

L. Gillick and S. Cox. Some statistical issues in the compairson of speech recognition algorithms. In *International Conference on Acoustics Speech and Signal Processing*, volume 1, pages 532–535, 1989.

J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proc. NAACL 2001*. ACL, 2001.

J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242, 1992.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML-01*, pages 282–289, 2001.

R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. CoNLL-2002*, 2002.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML 2000*, pages 591–598, Stanford, California, 2000.

T. P. Minka. Algorithms for maximum-likelihood logistic regression. Technical Report 758, CMU Statistics Department, 2001.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS 13*, pages 995–1001. MIT Press, 2001.

L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Proc. Third Workshop on Very Large Corpora*. ACL, 1995.

A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proc. EMNLP*, New Brunswick, New Jersey, 1996. ACL.

A. Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. In C. Cardie and R. Weischedel, editors, *EMNLP-2*. ACL, 1997.

S. Riezler, T. H. King, R. M. Kaplan, R. Crouch, J. T. Maxwell III, and M. Johnson. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proc. 40th ACL*, 2002.

E. F. T. K. Sang. Memory-based shallow parsing. *Journal of Machine Learning Research*, 2:559–594, 2002.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. URL `http://www-2.cs.cmu.edu/~jrs/jrspapers.html#cg`.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. CoNLL-2000*, pages 127–132, 2000.

H. Wallach. Efficient training of conditional random fields. In *Proc. 6th Annual CLUK Research Colloquium*, 2002.

A. Yeh. More accurate tests for the statistical significance of result differences. In *COLING-2000*, pages 947–953, Saarbruecken, Germany, 2000.

T. Zhang, F. Damerau, and D. Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 2002.